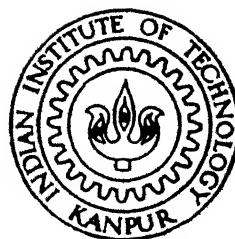


A Window based Package for Queueing Network Analysis

by

UMESH. M. N.

TH
EE/1997/m
Um N



DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

FEBRUARY 1997

A Window based Package for Queueing Network Analysis

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Umesh.M.N

to the

Department of Electrical Engineering

Indian Institute of Technology, Kanpur

February, 1997.

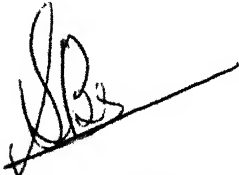
19 MAR 1997
CENTRAL LIBRARY
I. I. T., KANPUR

No. A 123243,

EE-1997-M-UME-WIN

CERTIFICATE

It is certified that the work contained in the thesis entitled **A Window based Package for Queueing Network Analysis** by **Umesh.M.N**, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.



Dr. S. K. Bose

Professor

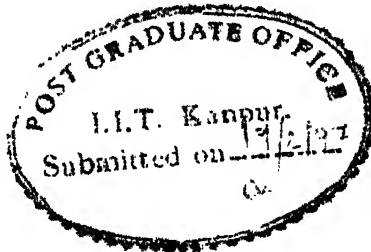
Department of Electrical Engineering
Indian Institute of Technology, Kanpur.



Dr. D. Manjunath

Assistant Professor

Department of Electrical Engineering
Indian Institute of Technology, Kanpur.



Acknowledgements

My sincere thanks to Dr.S.K.Bose and Dr.D.Manjunath for the excellent guidance I received from them during the course of this thesis work. Their words of support and encouragement have helped me reach this thesis to the present state.

I would like to express my deep felt gratitude to all in the Telematics and ERNET Labs, with special mention to Mrs.Sandhya Sule for the wholehearted co-operation extended to me during this work.

Life in the Lab would have been quite monotonous without my Telecom friends who have made it a pleasant one.

Finally, all my Mallu friends - Nimesh,Davis,Biju,Murali,Jacob and others deserve a word of gratitude for having made my days in IIT, something to remember for a long time.

Umesh.M.N

Abstract

Queueing Networks have been widely used to model and analyse the performance of complex waiting systems like communication networks, multiprogramming systems and even production job shops and vehicular traffic. Several algorithms exist for solving different types of queueing networks. This thesis deals with the design and implementation of a window based package for the analysis of queueing networks. Two of the well known algorithms - the GI/G/m technique and Mean Value Analysis have been incorporated into the package and have been tested for a wide variety of inputs. We have also developed the framework for further modifications of the package by leaving suitable provisions to include the other solution techniques as well.

Contents

List of Figures	iii
List of Tables	iv
1 Introduction	1
2 The GI/G/m Approximation	3
2.1 Introduction	3
2.2 Input Parameters	4
2.3 Network reconfiguration by immediate feedback removal	9
2.4 Internal flow parameter calculation	11
2.4.1 Internal arrival rate, λ_i	11
2.4.2 Internal arrival variance coefficient, c_{ai}^2	12
2.5 Output parameters	16
2.5.1 Node parameters	16
2.5.2 Class parameters	17
3 Mean Value Analysis	18
3.1 Introduction	18
3.2 Input Parameters	19
3.3 Internal Flow Rate Calculation	20
3.4 Mean Value Theorem	21
3.5 Algorithm	23
3.5.1 Single Class	23
3.5.2 Multiple class	25

3.6	Ouput Parameters	27
4	Implementation Details	28
4.1	The Analysis program	28
4.2	The User Interface program	29
4.3	C- <i>Mathematica</i> Interface	31
5	Results and Conclusion	32
5.1	Results	32
5.2	Conclusion	41
A	How to use the <i>Network Analysis & Simulation Tool</i>	42
A.1	Create a simple network	42
A.2	Modify the network	45
A.3	Input/Output procedure	47
A.4	Run the analysis / simulation program	48
A.5	File operations	48
	Bibliography	50

List of Figures

2.1	Superposition and Splitting.	3
2.2	External Arrival rate to Node i.	6
2.3	Arrival Rate from Node i to Node j.	7
2.4	Transition probability.	8
2.5	Immediate Feedback.	9
2.6	Internal Arrival Rate.	11
3.1	Equivalent Network as seen from Node 1.	22
5.1	Example 2.	33
5.2	Example 3.	34
5.3	Example 4.	36
5.4	Example 5.	37
5.5	Example 6.	38

List of Tables

5.1	Ouput Parameters for Example 1	33
5.2	Ouput Parameters for Example 2	34
5.3	Input Parameters for Example 3	34
5.4	Output Parameters for Example 3 (NAST)	35
5.5	Output Parameters for Example 3 (M/M/1)	35
5.6	Output Parameters for Example 4	36
5.7	Output Parameters for Example 5	37
5.8	Route Parameters for Example 6	37
5.9	Output Parameters for Example 6 (Node Outputs)	38
5.10	Output Parameters for Example 6 (Class Outputs)	38
5.11	Transition probability Matrix for Example 6	38
5.12	Route parameters for Example 7	39
5.13	Output Parameters for Example 7 (Node Outputs)	40
5.14	Output Parameters for Example 7 (Class Outputs)	40

Chapter 1

Introduction

Queueing Networks are widely used to model and analyse the performance of complex waiting systems like communication networks, multiprogramming systems and even production job shops. Their widespread acceptance is attributed to the fact that these models capture most of the characteristic features of the actual system. For example, a computer network with N nodes and packets transmitted from one node to another can be modelled as a network of queues. Parameters of the actual system like buffer space in each node, link capacities etc. have their equivalent representations in this model. Thus, solving the corresponding model using some realistic assumptions, if necessary, appears to be a better approach than direct measurement of the performance parameters.

Traditionally, the two techniques used to solve such a model are *analysis* and *simulation*. An analytic solution is based on a set of equations relating the input parameters to the performance measures. In some cases, however, we may not get such a set of equations; or the set may not be analytically solvable even if we can get one. In that case, a simulation approach may be used. Very often, simulation results are also used to validate the assumptions made in an approximate analysis of the same system.

Queueing Networks can be broadly classified into two types - *product form* and *non-product form* networks. Some of the commonly used queue types like First Come First Served (with exponential service times), Last Come First Served Preemptive Resume (with Coxian service times) etc. have the special property that the joint probability of the queue sizes in the network is just the product of individual queue length probability distributions with a normalization constant[1]. Networks of such queues are called

Product Form networks. Solving such networks is relatively simpler since we can analyse the queues separately and then substitute the results in the product form equation. Those networks which do not satisfy this property are collectively called Non-Product Form networks.

Several algorithms have been developed to solve either type of networks. This thesis deals with the design and implementation of a graphical tool for the analysis/simulation of queueing networks. A wide variety of such packages are commercially available viz. QNA, PANACEA, ResQ, BEST/1, CADS and so on. These packages, in general rely upon a single algorithm for the analysis. However, the requirements of a typical user may not always be satisfied by one method alone. This arises from the fact that it would reduce the flexibility of usage by imposing certain constraints on factors like the input parameter set which one should have to use with the algorithm, the list of output measures that one can have etc. Our package has been developed keeping in mind such difficulties, by allowing the user to select the preferred solution algorithm. Most of the widely available queueing network solution techniques, meant for different kinds of queues can therefore be incorporated into this. The tool has the added advantage of a simulation option if required. The interface to the analysis/simulation program is through a window based frontend, giving a user friendly GUI.

A brief description of the contents of this thesis is as follows. Chapter 2 explains an approximate solution technique for queueing networks with general arrival and service rates. This algorithm does not make any assumptions about the network to be solved and hence can be used for both product form as well as non-product form networks. Chapter 3 discusses a well-known algorithm for closed product form networks, called the Mean Value Analysis. The implementation of this algorithm is comparatively easier than the first one. Chapter 4 gives a brief idea regarding the implementation of these two algorithms and that of the graphical user interface. Several solution techniques exist for various other types of queues which can be incorporated into this package depending on their applicability. This would be done as a future work as mentioned in the concluding chapter. Appendix A is a manual giving details regarding how the package can be used.

Chapter 2

The GI/G/m Approximation

2.1 Introduction

The class of queueing network solution algorithms in which individual nodes are analysed in isolation, based on the input and output processes of each, is known as *Parametric Decomposition*[2]. The GI/G/m approximation method that is used in our *Network Analysis and Simulation Tool* is a solution technique of this type.

The approximations made by this approach is such that we require only the mean and the squared coefficient of variance ($variance/mean^2$) of the interarrival time and service time distributions for our computations. The queueing network that is to be solved may have nodes at which customer streams are split, superposed or both. This implies that the internal arrival process to the nodes is different from the external arrival into it. An example for this has been shown in Fig 2.1

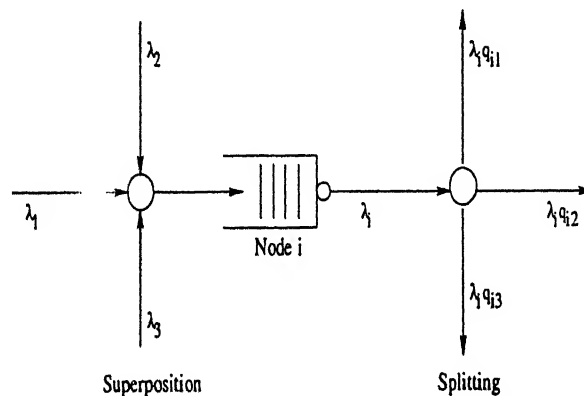


Figure 2.1: Superposition and Splitting.

The algorithm requires the arrival process to a node to be renewal in nature in which the arrival intervals are all i. i. d random variables. In certain situations, however, processes which are not renewal also need to be approximated as a suitable renewal process. The algorithm first calculates the mean and variance of the internal arrival process to individual nodes by solving a set of linear equations. These equations have been generated using approximations for the resulting process in each of the three situations where traffic is split, superposed or both split and superposed. Then it replaces the internal arrival process by a renewal process with equivalent first and second moments. In this way, the queueing network is decomposed into individual subsystems which can then be analysed separately. Each subsystem is a standard GI/G/m queue characterised by the first two moments of its arrival process and service time. Approximate two-moment formulae are available for the congestion measures in such queues. These are used in deriving the required output parameters[3].

Some of the assumptions made by the algorithm are as follows.

- The network is open rather than closed.
- A queue can have one or more servers and has infinite waiting space.
- The service discipline is FCFS.
- There can be different classes of customers but a customer cannot change classes within the network.
- There can be creation or combination of customers at each node.
- The customer routing matrix is static.

A detailed description of the algorithm is given in the following sections.

2.2 Input Parameters

The algorithm for analysing the GI/G/m queue used in this package can handle both single and multiple classes of customers. For the single class case only five parameters per node are required in addition to the transition probability matrix. For the multiple class model a larger set of input parameters is required. These are subsequently

converted into the single class input type. Each of these input options is discussed below.

Single Class Input

This option requires the following input data to be provided-

n	no. of nodes;
m_i	no. of servers in node i ;
λ_{0i}	external arrival rate to node i ;
c_{0i}^2	squared coeff. of variation of the external arrival to node i ;
τ_i	average service time at node i ;
c_{si}^2	squared coeff. of variation of service time pdf at node i ;
$Q = [q_{ij}]$	Transition probability matrix;
ν_i	multiplication factor of node i .

The multiplication factor ν_i takes care of any creation or combination of customers at node i . For example, a message to be sent by a node may be split into small packets and transmitted. Later, at the destination these packets are recombined to form the entire message. Such situations can be best modelled by the multiplication factor.

Multiple Class Input

In the mutple class option, each class has a fixed routing list of its own. This list contains the set of nodes visited by a class. Note that this is unlike the single class case where we had a probabilistic routing table. An important point to be remembered here is that the route for a particular class i starts at some node and teminates at another node. In between this route there is no further inflow of class i at any node so that the arrival rate of class i to any node in its route remains the same.

The set of inputs for this option are given here.

n	no. of nodes;
m_i	no. of servers at node i ;
ν_i	multiplication factor of node i ;
r	no. of classes.

For each class(say class k) the parameters required are-

- n_k no. of nodes in its route;
- λ_k external arrival rate;
- c_k^2 squared coeff. of variation of ext. arrival process;
- τ_{kj} average service time at the j th node in its route;
- c_{skj}^2 squared coeff. of variation of service time at the j th node in its route;
- $n_{k,j}$ j th node in its route.

This input set is now transformed into the single class input form using the relations that are given below.

Note: In the following discussions, $I_W(\omega)$ is defined as the indicator function of W such that

$$I_W(\omega) = \begin{cases} 1 & \text{if } \omega \in W \\ 0 & \text{otherwise} \end{cases}$$

1. External arrival rate to node i , λ_i :

We know that the net arrival rate into node i is the sum of the external arrival rates λ_k of all the classes whose routing list begins with node i .

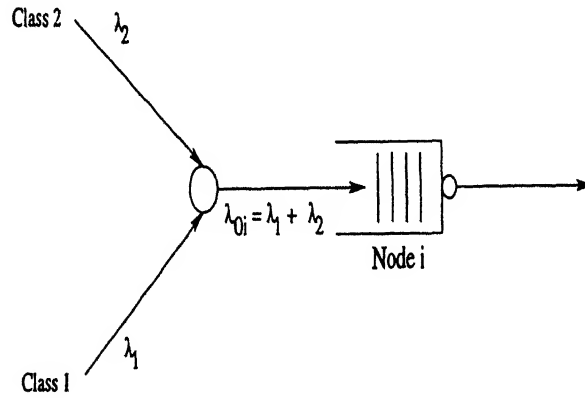


Figure 2.2: External Arrival rate to Node i .

Mathematically,

$$\lambda_{oi} = \sum_{k=1}^r \lambda_k I_K(k)$$

where $K = \{m : n_{m,1} = i\}$

2. Arrival rate from i to j, λ_{ij} :

This is calculated in the same way as above and is given by

$$\lambda_{ij} = \sum_{k=1}^r \sum_{l=1}^{n_k-1} \lambda_k I_{K_L}(k)$$

where $K_L = \{(p, q) : n_{p,q} = i; n_{p,q+1} = j\}$

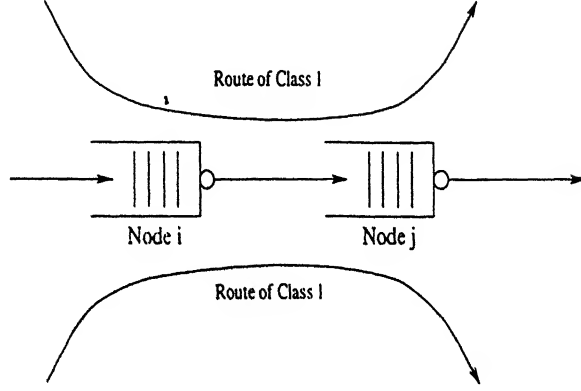


Figure 2.3: Arrival Rate from Node i to Node j.

The first summation is from 1 to $(n_k - 1)$ since the last node in a route need not be considered. As clear from Fig 2.3, λ_{ij} is the sum of the external arrival rates of all the routes (classes) for which j is the immediate successor of i.

3. Outflow from node i, λ_{i0} :

Similar to the above cases, the rate at which customers depart from the network at node i is calculated as the sum of the external arrival rates of all classes which have node i as the last node.

$$\lambda_{i0} = \sum_{k=1}^r \lambda_k I_K(k)$$

where $K = \{m : n_{m,n_m} = i\}$

4. Transition Matrix, Q:

The transition probability matrix is calculated from λ_{i0} and λ_{ij} values obtained previously. In particular,

$$q_{ij} = \frac{\lambda_{ij}}{\lambda_{i0} + \sum_{k=1}^n \lambda_{ik}}$$

that is the ratio of net outflow from i that goes to j.

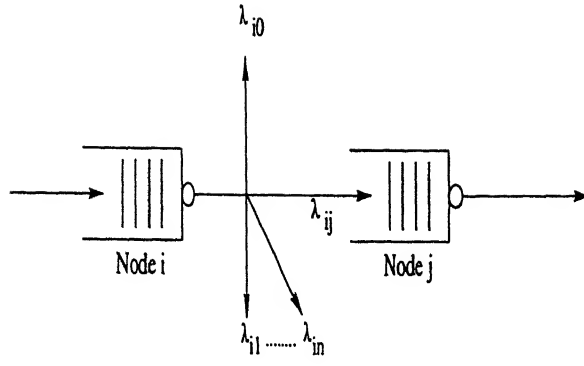


Figure 2.4: Transition probability.

5. **Average service time, τ_i :**

This is determined as the average of the service times (at node i) of all the classes which have their route passing through i. Hence,

$$\tau_i = \frac{\sum_{k=1}^r \sum_{l=1}^{n_k} \lambda_k \tau_{kl} I_{K_L}(k, l)}{\sum_{k=1}^r \lambda_k I_{K_L}(k, l)}$$

where $K_L = \{(m, n) : n_{m,n} = i\}$

For example, consider a 3-node, 3-class network, for which the routing list of individual classes is given below.

Class 1: {2, 1, 3}

Class 2: {1, 3, 1}

Class 3: {3, 2, 1}

Hence the average service time of Node 2 is calculated as,

$$\tau_2 = \frac{\lambda_1 \tau_{11} + \lambda_3 \tau_{32}}{\lambda_1 + \lambda_3}$$

6. **Service time variance coefficient at node i, c_{si}^2 :**

This is calculated as in the case of τ_i , except that the second moment of the service time distribution is used instead of the mean.

$$\tau_i^2 (c_{si}^2 + 1) = \frac{\sum_{k=1}^r \sum_{l=1}^{n_k} \lambda_k \tau_{kl}^2 (c_{skl}^2 + 1) I_{K_L}(k, l)}{\sum_{k=1}^r \sum_{l=1}^{n_k} \lambda_k I_{K_L}(k, l)}$$

where $K_L = \{(m, n) : n_{m,n} = i\}$

7. **External arrival variance coefficient at node i, c_{0i}^2 :**

The last parameter to be calculated is the variance coefficient of the external arrival

process to node i . If λ_{0i} is 0 this parameter is of no consequence to us and is set to 1. Otherwise, it is derived using approximations for the variance in the case of superposition of arrival processes.

$$c_{0i}^2 = (1 - w_i) + w_i \left[\sum_{k=1}^r c_k^2 \left(\frac{\lambda_k I_K(k)}{\sum_{l=1}^r \lambda_l I_K(l)} \right) \right]$$

where

$$w_i = \left[1 + 4(1 - \rho_i)^2 (\gamma_i - 1) \right]^{-1}$$

$$\gamma_i = \left[\sum_{k=1}^r \left(\frac{\lambda_k I_K(k)}{\sum_{l=1}^r \lambda_l I_L(l)} \right)^2 \right]^{-1}$$

$$K = L = \{m : n_{m,1} = i\}$$

Some additional details regarding the derivation of this result are given in Section 2.4.2

2.3 Network reconfiguration by immediate feedback removal

In the GI/G/m algorithm it is presumed that the input and output processes to a node are uncorrelated with each other. This assumption would be violated in the case of immediate feedback, where $q_{ii} > 0$. The errors in the approximation were found to be beyond tolerable limits in such cases[2]. Hence, the network has to be modified in such a way that the immediate feedback is eliminated without affecting the other parameters.

Consider a queue with immediate feedback as shown in Fig 2.5.

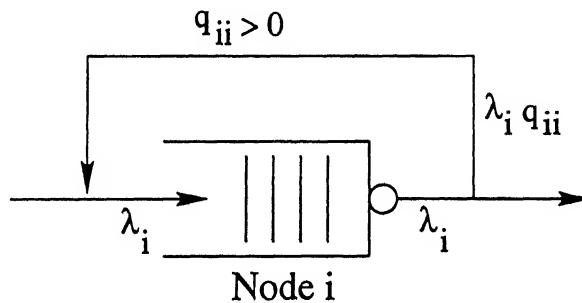


Figure 2.5: Immediate Feedback.

Let $H_i(s)$ be the transform of the service time distribution of this queue. A customer who has finished service, returns to the end of the queue with probability q_{ii} so that he gets served X times (where X is a random variable) before he leaves the queue. Hence the total service time of a customer is given as

$$xH_i(s)$$

where the customer returns to the queue $X = x$ times.

Also,

$$P_x = q_{ii}^{x-1}(1 - q_{ii})$$

where P_x denotes the probability that the customer gets served x times before leaving the system. Therefore, the total service time has a transfer function

$$\begin{aligned} H(s) &= \sum_{x=1}^{\infty} [xH_i(s)] P_x \\ &= \frac{(1 - q_{ii})H_i(s)}{1 - q_{ii}H_i(s)} \end{aligned}$$

From this the modified service parameters can be calculated as,

$$\begin{aligned} \tau'_i &= \frac{\tau_i}{(1 - q_{ii})} \\ c_{si}^{\prime 2} &= q_{ii} + (1 - q_{ii})c_{si}^2 \end{aligned}$$

In addition, the transition probabilities have to be modified as

$$\begin{aligned} q'_{ii} &= 0 \\ q'_{ij} &= \frac{q_{ij}}{1 - q_{ii}} \end{aligned}$$

The essence of this reconfiguration is that it combines all the service phases of a customer in a queue into a single large service period. Afterwards, when we calculate the congestion measures we get the values for this aggregate service period and not for the individual service times. To get the parameters for one visit to a node, we have to modify the output values accordingly, as follows.

$$\begin{aligned} W_i &= (1 - q_{ii})W'_i \\ Var(W_i) &= (1 - q_{ii}) + \psi(N_i) - c_{si}^{\prime 2}\tau_i^{\prime 2} \end{aligned}$$

where

- W_i is the average waiting time in the modified network;
- W'_i is the average waititng time in the original network;
- N_i is the average queue size including the customers in the server, and
- $\psi()$ is a function of N_i

The first equation is self-explanatory since a customer makes an average of $(1 - q_{ii})^{-1}$ immediate visits to node i, before leaving it. The second result is based on heavy traffic limits for queues[4] .It is quite obvious that the feedback elimination does not affect the mean number in the node.

2.4 Internal flow parameter calculation

Once the external flow moments have been obtained as explained in the preceding sections, we can calculate the mean, λ_i and variance coefficient, c_{ai}^2 of the internal flow process to node i.

2.4.1 Internal arrival rate, λ_i

In this step we make use of the flow conservation rule at node i which says that the departure rate from the node is the same as the arrival rate into it.

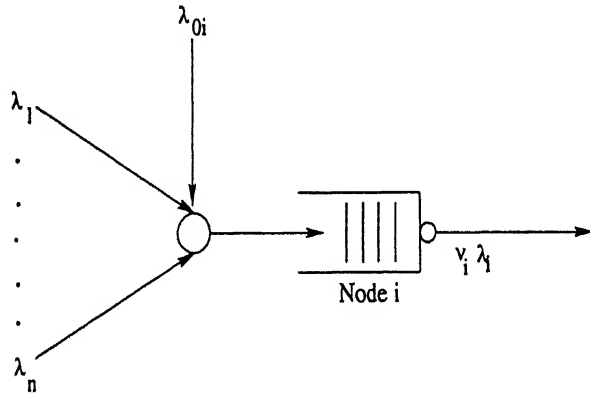


Figure 2.6: Internal Arrival Rate.

Hence the final departure rate is equal to ν_i times λ_i where ν_i is the multiplication factor of node i.

At the input to node i , we have a superposition of $(n + 1)$ arrival processes and the mean of the resultant process is λ_i . Since the average of a sum of processes is just the sum of averages of individual ones, we get

$$\lambda_i = \lambda_{0i} + \sum_{j=1}^n \lambda_j \nu_j q_{ji}$$

or in matrix notation as

$$\Lambda = \Lambda_0 + \Lambda \Gamma Q$$

Hence,

$$\Lambda = \Lambda_0 (I - \Gamma Q)^{-1}$$

Note: Eventhough we know that $q_{ij} = 0$ for $i = j$ (as a result of the reconfiguration of the network) we include this condition also, in the above equation for ease of calculations.

The solution of this matrix equation gives the internal arrival rates, using which the utilisation at node i is calculated as

$$\rho_i = \frac{\lambda_i \tau_i}{m_i}$$

2.4.2 Internal arrival variance coefficient, c_{ai}^2

Here we again get a system of linear equations but the derivation is not as straight forward as in the above case. This is because, the superposition of k arrival processes (which we assume to be renewal) does not yield another renewal process, except when each of the component process is Poisson. The implication here is that we cannot get the n th moment of the superposition stream by simple addition of the n th moments of individual streams ($n > 1$). Hence we have to resort to approximations for deriving the variance and higher moments.

Superposition of arrival processes

Consider the internal flow process into node i . As clear from the above explanation, this is a point process (in which successive arrival intervals are i. i. d). Our attempt here is to replace this by a renewal process, with the same moment characteristics. Since equating all the moments would not be possible we use only the mean and variance.

The basic principle is to use the n th partial sum, S_n (which is the sum of the first n arrival intervals) of the superposition process and use its first two moments to define the approximating renewal process[5].

Let $M_j(X)$ denote the j th moment of some random variable X . Now, if H represents the interarrival time c. d. f of the approximating renewal process, the moments of H are calculated from those of the n th partial sum to be

$$M_j(H) = \frac{M_j(S_n)}{n}$$

In other words, we first calculate the moments of S_n and then divide it by n to get the approximate moments of H .

There are two methods which are commonly utilised for this substitution. In the first one, called *stationary-interval method*, we use $n = 1$; i. e the moments of H are calculated from those of the first arrival interval of the point process. Though this method does not take into consideration the interdependence among subsequent intervals, this is reasonable in many cases.

In the second technique, the *asymptotic method*, we consider the asymptotic behaviour of the point process as time tends to infinity. Thus we use $n = \infty$ to get the moments of H .

In general, any one method alone does not work well for the entire range of traffic intensities. Intuitively, we can see that the stationary-interval method should work well for low traffic intensities, since the interarrival time would be large enough to reduce the approximation errors. However the asymptotic method is found to be more accurate under heavy traffic conditions. Hence a hybrid combination of the moments calculated using both the methods is used as a consensus.

However, there is one problem with the hybrid which makes it unsuitable for our applications. The stationary-interval method, when used to approximate a superposition stream by a renewal process gives a variance, which is a non-linear function of the variances of component processes. This would lead to a non-linear set of equations to be solved to deduce the second moment of the internal arrival process. Fortunately, a convex combination of the variance obtained by asymptotic method and that of an exponential (which has variance coefficient 1) is found to perform equally well[6] and hence this is used in the algorithm.

Derivation of c_{ai}^2

Let us consider the departure process of node j . To approximate this point process by a renewal process we first use the stationary-interval method which gives[3],

$$(c_{dj}^2)_S = \begin{cases} \rho_j^2 c_{sj}^2 + (1 - \rho_j^2) c_{aj}^2 & \text{for GI/G/1 queue} \\ 1 + (1 - \rho_j^2)(c_{aj}^2 - 1) + \rho_j^2 m_j^{-0.5}(\max\{c_{sj}^2, 0.2\} - 1) & \text{for GI/G/m queue} \end{cases}$$

where the subscript S denotes stationary-interval.

We know that the number of departures d_t in time t is equal to the number of arrivals a_t in the same interval minus the number in the queue n_t at time t . i. e

$$d_t = a_t - n_t$$

With t tending towards ∞ , as in the asymptotic approximation, n_t tends to a steady state value (assuming stability conditions for the queue) and hence d_t would follow a_t . In other words, the asymptotic approximation of c_{dj}^2 is c_{aj}^2 itself.

$$(c_{dj}^2)_A = c_{aj}^2$$

where the subscript A denotes asymptotic.

At the output of node j , the stream is split according to q_{ji} . If a renewal process with variance coefficient c^2 is split into k streams, according to probability p_i , the variance coefficient of the i th stream can be derived as

$$c_i^2 = p_i c^2 + (1 - p_i)$$

Thus, for the stationary-interval method,

$$(c_{ji}^2)_S = q_{ji} (c_{dj}^2)_S + (1 - q_{ji})$$

and for asymptotic method,

$$(c_{ji}^2)_A = q_{ji} (c_{dj}^2)_A + (1 - q_{ji})$$

Hence, the variance coefficient of the stream from j to i , as a hybrid of the above two is,

$$c_{ji}^2 = \alpha_{ji} (c_{ji}^2)_A + (1 - \alpha_{ji}) (c_{ji}^2)_S$$

Thus, we have replaced the arrival streams into individual nodes by a renewal process. It has to be noted, however, that the departure process that is being split is not really renewal and hence, there is some error associated with the result.

Now, the internal arrival process to each node is the superposition of these renewal processes. As explained in the last section, we use a convex combination of the asymptotic value and an exponential variance coefficient of 1, to get the variance coefficient of this process.

The asymptotic variance coefficient can be found to be

$$\begin{aligned} (c_{ai}^2)_A &= \sum_{j=0}^n \left(\frac{\lambda_{ji}}{\sum_{k=0}^n \lambda_{jk}} \right) c_{ji}^2 \\ &= \sum_{j=0}^n p_{ji} c_{ji}^2 \\ p_{ji} &= \frac{\lambda_{ji}}{\sum_{k=0}^n \lambda_{jk}} \end{aligned}$$

Thus, p_{ji} represents the proportion of arrivals to i that come from j . Now,

$$c_{ai}^2 = \omega_i (c_{ai}^2)_A + (1 - \omega_i)$$

This can be written in the form,

$$c_{ai}^2 = u_i + \sum_{j=1}^n c_{aj}^2 v_{ji}$$

where

$$\begin{aligned} u_i &= 1 + w_i \left[(p_{0i} c_{0i}^2 - 1) + \sum_{j=1}^n p_{ji} \{ (1 - q_{ji}) + \nu_j q_{ji} \rho_j^2 x_j \} \right] \\ v_{ji} &= w_i p_{ji} q_{ji} \nu_j (1 - \rho_j^2) \\ x_j &= 1 + m_j^{-0.5} [\max(c_{sj}^2, 0.2) - 1] \\ w_i &= \frac{1}{\{ 1 + 4(1 - \rho_i)^2 (\gamma_i - 1) \}} \\ \gamma_i &= \frac{1}{\sum_{i=0}^n p_{ij}^2} \end{aligned}$$

This is a set of linear equations and can be solved to get c_{ai}^2 . Thus, we have derived all the parameters required to calculate the congestion measures at the nodes.

2.5 Output parameters

2.5.1 Node parameters

1. Waiting time moments:

To calculate the average waiting time W_i , for a GI/G/1 queue we use the Kraemer and Langenbach-Belz approximations given as,

$$W_i = \frac{\tau_i \rho_i (c_{ai}^2 + c_{si}^2) \beta}{2(1 - \rho_i)}$$

where

$$\beta = \begin{cases} e^{\left[-\frac{2(1-\rho_i)(1-c_{ai}^2)^2}{3\rho_i(c_{ai}^2 + c_{si}^2)} \right]} & c_{ai}^2 < 1 \\ 1 & \text{otherwise} \end{cases}$$

If the number of servers is more than 1, we use another result based on the heavy traffic limit theorems, where

$$W_i = \left(\frac{c_{ai}^2 + c_{si}^2}{2} \right) W_i^{M/M/m}$$

The variance coefficient of the waiting time is approximated to be the same as that of an M/M/m queue with the same set of input parameters.

2. Queue size parameters:

The average number in the system, N_i can be found out from Little's formula, as

$$N_i = \rho_i m_i + \lambda_i W_i$$

The variance coefficient does not vary too much from the corresponding M/M/m value. The average number in the whole network is the sum of the mean queue sizes at the individual nodes. Similarly the variance coefficient for this is approximately the same as the sum of variances.

3. Departure rates:

If the multiplication factor ν_i of all the nodes are equal to 1, the total output flow rate from the network is equal to the net input flow rate. Hence, total departure rate from the network,

$$d = \sum_{i=1}^n \lambda_{0i}$$

In the general case, when $\nu_i \neq 1$ we have to consider the departure rate from individual nodes, to get

$$d = \sum_{i=1}^n \lambda_i \nu_i (1 - \sum_{j=1}^n q_{ij})$$

4. Visit ratios:

This is defined as the average number of visits to node i for a customer, during his time in the network and is given by,

$$V_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_{0j}}$$

5. Sojourn time:

The average time T_i that an arbitrary customer spends in node i , till he departs from the system is given as,

$$T_i = V_i(\tau_i + W_i)$$

and the expected total time in the whole network (also called the total sojourn time) is,

$$T = \sum_{i=1}^n T_i$$

2.5.2 Class parameters

This algorithm provides some output measures for individual classes as well. For class k ,

Average service time	$= \sum_{j=1}^{n_k} \tau_{kj}$
Variance coefficient of service time	$= \sum_{j=1}^{n_k} \tau_{kj}^2 c_{sj}^2$
Average waiting time	$= \sum_{j=1}^{n_k} W_{n_{k,j}}$
Variance coefficient of waiting time	$= \sum_{j=1}^{n_k} Var(W_{n_{k,j}})$
Sojourn time	$= \sum_{j=1}^{n_k} (\tau_{kj} + W_{n_{k,j}})$

Note that in writing the above relations, we have assumed that the congestion measures at each node in the route of a class are independent parameters. Otherwise the summation of the variances of the waiting time(service time) at each node would not be equal to the variance of the total waiting time(service time).

Chapter 3

Mean Value Analysis

3.1 Introduction

Parameters of a queue that are measurable during a finite observation period are called *operational variables*. These can either be directly measured or derived from other operational quantities. Let a_t be the number of arrivals, b_t be the total busy period and c_t be the number of service completions in an observation interval t . All of these are operational variables of the queue since they can be physically measured. For this time interval, we can derive the arrival rate λ_t , the departure rate d_t , the service rate μ_t and utilisation ρ_t from these basic quantities as,

$$\begin{aligned}\lambda_t &= \frac{a_t}{t} \\ d_t &= \frac{c_t}{t} \\ \rho_t &= \frac{b_t}{t} \\ \mu_t &= \frac{c_t}{b_t}\end{aligned}$$

Also, it can be seen that

$$\rho_t = \mu_t d_t$$

Such relations between operational variables which hold in any observation period are called *Operational Laws*. In fact, the well-known Little's formula is also an operational law of this kind.

The Mean Value Analysis(MVA) technique described in this chapter solves a product form closed queueing network on the basis of a few operational laws. The MVA is an iterative procedure which calculates the throughput,waiting time and average number in each queue. We cannot calculate the higher moments of the system parameters with this algorithm. However, MVA is still preferred due to the simplicity of its implementation.

3.2 Input Parameters

The MVA approach may be used for both single and multiple customer classes. Unlike the GI/G/m algorithm described in Chapter 2, no conversion of the multiple class inputs to a single class form is required.

Note: We have implemented only the Single class algorithm in the present version of *Network Analysis & Simulation Tool*. However, we include the details of Multiple class MVA also in this chapter, for the sake of completeness.

Single Class Input

The single class option requires the following input data to be provided-

- n number of nodes;
- C number of customers circulating in the network;
- μ_i service rate of node i;
- m_i number of servers at node i;
- $Q = [q_{ij}]$ Transition probability matrix

In addition to these, we also require the service discipline at each node. This has to be one of the following :

- First Come,First Served(FCFS)
- Last Come,First Served Pre-emptive Resume(LCFS-PR)
- Infinite Server(IS)
- Processor Sharing(PS)

Baskett,Chandy,Muntz and Palacios[1] have shown that a product form solution for a network exists if and only if the service discipline of each queue is one of the above four. For

an FCFS queue, the service time distribution must be exponential. For the other cases the service time should have a probability distribution with a rational Laplace Transform.

Multiple Class Input

For the MVA Multiple class, we require the following inputs-

- n number of nodes;
- m_i number of servers at node i;
- R number of classes;

For each class (say Class k), the parameters required are-

- μ_{mk} Service rate at node m;
- C_k Number of customers in the network;
- $Q = [q_{ijk}]$ Transition Probability Matrix for Class k;

As in the Single class option, we also require the service discipline at each node, which has to be FCFS, LCFS-PR, IS or PS.

3.3 Internal Flow Rate Calculation

We calculate the internal flow rate into node i, in a manner similar to the GI/G/m analysis. However, in this case we have a set of dependent equations since we are considering a closed network.

Thus we have, for Single class MVA,

$$\lambda_i = \sum_{j=1}^n \lambda_j q_{ji}$$

The algorithm requires only the visit ratio at each node(which is defined as the average number of visits made to node i, for each visit to a reference node, say node 1) and not the actual numerical values of the arrival rates. Hence, we solve the above set of equations assuming a value of 1 for λ_1 . The solution gives the visit ratio to each node with respect to node 1.

The internal arrival rate calculation for Multiple class is similar to the above case, except that we calculate the visit ratio at each node in the route of a class. The flow balance equation can be modified as,

$$\lambda_{ik} = \sum_{j=1}^n \lambda_{jk} q_{jik}$$

3.4 Mean Value Theorem

In this section, we give a brief explanation of the principles used in Mean Value Analysis. The Single class case alone is considered for ease of understanding.

The basic equation in the iteration has been obtained from the Mean Value Theorem formulated by Reiser and Lavenberg[7]. The theorem states that a customer arriving to a queue in a product form network sees precisely the same average number in the queue as an outside observer would see, if the network had one less customer. Let $N_i(c)$ be the average size of queue i when c customers are circulating in the network, and $W_i(c)$ be the average waiting time in the queue. If we have a load independent server, the Mean Value Theorem gives a simple relation between $W_i(c)$ and $N_i(c-1)$.

$$W_i(c) = \frac{1 + N_i(c-1)}{\mu_i} \quad (3.1)$$

Note that this equation is valid only in the case of single server FCFS, LCFS-PR and PS queues. Obviously, in the case of an IS queue

$$W_i(c) = \mu_i^{-1} \quad (3.2)$$

This is the recurrence relation that is used in the algorithm. It has to be noted, however, that the above relation is not an operational law since it assumes memoryless service, a condition which cannot be physically tested.

For a FCFS queue with multiple servers, the above relation has to be modified since we have to consider the occupation probabilities at each node as well. Let m_i be the number of servers at node i . The average waiting time when there are c customers in the network can be obtained as a special case of the Mean Value Theorem.

$$W_i(c) = \frac{1}{\mu_i m_i} \left\{ 1 + N_i(c-1) + \sum_{j=1}^{m_i-1} (m_i - j) P_i \{j-1, c-1\} \right\} \quad (3.3)$$

where $P_i \{j - 1, c - 1\}$ is the probability of having $j - 1$ customers in queue i when $c - 1$ customers are present in the closed network. This calculation is more complex than the single server case due to the probability factor occurring in the equation. A recurrence equation for $P_i \{j, c\}$ can also be derived as,

$$P_i \{j, c\} = \begin{cases} \lambda_1(c) P_i \{j - 1, c - 1\} \mu_i^{-1} & \text{for } j=1 \dots n \\ 1 - \sum_{k=1}^n P_i \{k, c\} & \text{for } j=0 \end{cases} \quad (3.4)$$

Thus we now have an equation which relates the average waiting time in a closed queue with c customers to the average number in the queue when there are $c - 1$ customers. Hence, if we had another relation to update the value of $N_i(c)$ we would get an iterative procedure to determine the average parameter values when there are C customers in the network. This equation is provided by Little's theorem.

Consider Node 1 in the network, whose internal arrival rate is λ_1 . We have assigned this as the reference node for the visit ratio calculations. That is, for any node k in the network

$$V_k = \frac{\lambda_k}{\lambda_1}$$

where V_k is the visit ratio to node k .

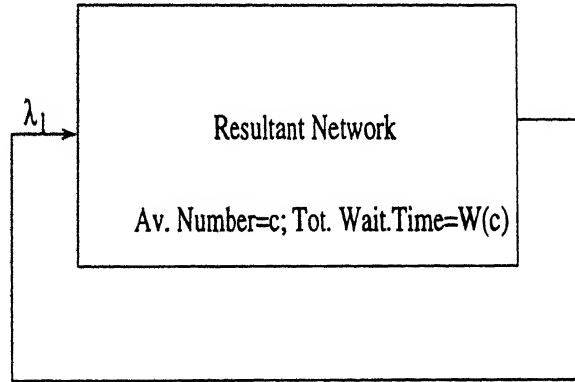


Figure 3.1: Equivalent Network as seen from Node 1.

The total number in the closed network shown in Figure 3.1 is c , and the total waiting time $W(c)$ is the sum of the average waiting time in each queue in the network. Note that we consider here the values as seen by a customer entering node 1.

$$W(c) = \sum_{i=1}^n W_i(c) V_i \quad (3.5)$$

where W_i is as defined earlier for each service discipline. The visit ratio has been included in the above summation to take into account multiple visits to the same node. Hence, applying Little's result we get

$$\lambda_1 = \frac{c}{W_c} \quad (3.6)$$

This new value of λ_1 is used to modify the queue lengths at each node, when there are c customers in the network. For individual queues

$$N_i(c) = \lambda_1 V_i W_i(c) \quad (3.7)$$

The updated values of $N_i(c)$ are used in the next iteration. The algorithm is described in a more formal manner in the next section.

3.5 Algorithm

3.5.1 Single Class

As clear from the above section, the algorithm consists of three parts. First, we calculate the average waiting time in each queue with c customers from the average number in each queue, when there where $c - 1$ customers in the network. (Refer Equations 3.1 to 3.3).

Next, we consider the system as seen by a customer entering the reference node, i. e Node 1, and calculate the throughput of the system using Equation 3.5 and 3.6 Note that the throughput calculated in this step is nothing but the internal arrival rate λ_1 to Node 1.

We know that the average number in a queue is the product of its arrival rate and the average waiting time. Also, by the definition of visit ratio, the average arrival rate to any node in the network is given as the arrival rate to Node 1 multiplied by the visit ratio of the node. Using the results of Equations 3.1 to 3.6 and applying Little's result we update the average number in each queue in the last step. We state the computational procedure below in algorithmic notation:

1. Initialisation

$$N_i(0) = 0; P_i \{0, 0\} = 1; P_i \{j, 0\} = 0 \text{ for } i = 1 \dots n \text{ and } j = 1 \dots m_i - 1$$

2. Main loop

Repeat Steps 3...6 for $c = 1 \dots C$

3. Mean Value Theorem on each node

For $i = 1 \dots n$

$$W_i(c) = \begin{cases} (1 + N_i(c-1))\mu_i^{-1} & \text{for Single server FCFS, LCFS-PR and PS} \\ \mu_i^{-1} & \text{for IS} \\ \{1 + N_i(c-1) + S_i\} (\mu_i m_i)^{-1} & \text{for Multiple server FCFS} \end{cases}$$

where $S_i = \sum_{j=1}^{m_i-1} (m_i - j) P_i \{j-1, c-1\}$

4. Little's Theorem for the network

At Node 1,

$$\begin{aligned} W(c) &= \sum_{i=1}^n W_i(c) V_i \\ \lambda_1 &= \frac{c}{W_c} \end{aligned}$$

5. Little's Theorem on each node

For $i = 1 \dots n$

$$N_i(c) = \lambda_1 V_i W_i(c)$$

6. Recurrence equation for probability update

At each Multiple server FCFS node i,

$$P_i \{j, c\} = \begin{cases} \lambda_1(c) P_i \{j-1, c-1\} (\mu_j)^{-1} & \text{for } j=1 \dots n \\ 1 - \sum_{k=1}^n P_i \{k, c\} & \text{for } j=0 \end{cases}$$

3.5.2 Multiple class

The single class MVA had a main loop which incremented the number of customers in each iteration. However, in the Multiple class case there are different classes of customers and hence we have a loop for each class.

As in the Single class case, we calculate first the average waiting time at Node i for Class k , W_{ik} from the average queue sizes obtained in the previous iteration, with one Class k customer less. The equations are obtained from the Mean Value Theorem and are given later.

In the second step, we determine the throughput (i. e the internal arrival rate at Node 1) for each class k , by applying Little's result to the equivalent network as seen by a Class k customer entering Node 1.

Finally, we update the queue length at each node by adding together the queue lengths of each class, which can be calculated from the results of the first two steps.

These steps have to be executed for all combinations of customer population. Note that in this case the population c is a vector given by,

$$c = \{c_1, \dots, c_R\}$$

where c_i denotes the population of Class i .

Let $W_{ik}(c)$ denote the average waiting time of Class k at Node i , V_{ik} be the visit ratio of Class k at Node i and e_k be an R -dimensional unit vector. We summarize the calculations in algorithmic notation:

1. Initialisation

$$N_i(0) = 0; P_i\{0, 0\} = 1; P_i\{j, 0\} = 0 \text{ for } i = 1 \dots n \text{ and } j = 1 \dots m_i - 1$$

2. Main loop

Repeat Steps 3...6 for $c_1 = 1 \dots C_1; c_2 = 1 \dots C_2; \dots; c_R = 1 \dots C_R$

3. Mean Value Theorem on each node for each class

For $i = 1 \dots n; k = 1 \dots R$ and if Node i is in the route of Class k ,

$$W_{ik}(\mathbf{c}) = \begin{cases} (1 + N_i(\mathbf{c} - \mathbf{e}_k))\mu_{ik}^{-1} & \text{for Single server FCFS, LCFS-PR and} \\ \mu_{ik}^{-1} & \text{for IS} \\ \{1 + N_i(\mathbf{c} - \mathbf{e}_k) + S_{ik}\}(\mu_{ik}m_i)^{-1} & \text{for Multiple server FCFS} \end{cases}$$

where $S_{ik} = \sum_{j=1}^{m_i-1} (m_i - j) P_i \{j - 1, \mathbf{c} - \mathbf{e}_k\}$

4. Little's Theorem for each class

At Node 1, for $k = 1 \dots R$

$$W_k(\mathbf{c}) = \sum_{i=1}^n W_{ik}(\mathbf{c}) V_{ik} I_k(i) I_k(i)$$

$$\lambda_{1k} = \frac{C_k}{W_k(\mathbf{c})}$$

where

$$I_k(i) = \begin{cases} 1 & \text{if Node } i \text{ is in the route of Class } k \\ 0 & \text{otherwise} \end{cases}$$

5. Little's Theorem on each node

For $i = 1 \dots n$

$$N_i(\mathbf{c}) = \sum_{k=1}^R \lambda_{1k} V_{ik} W_{ik}(\mathbf{c})$$

6. Recurrence equation for probability update

At each Multiple server FCFS node i ,

$$P_i \{j, \mathbf{c}\} = \begin{cases} \sum_{m=1}^R \lambda_{1m}(\mathbf{c}) P_i \{j - 1, \mathbf{c} - \mathbf{e}_m\} (\mu_{rj})^{-1} I_i(m) & \text{for } j=1, \dots, n \\ 1 - \sum_{k=1}^n P_i \{k, \mathbf{c}\} & \text{for } j=0 \end{cases}$$

where

$$I_i(m) = \begin{cases} 1 & \text{if Class } m \text{ passes through Node } i \\ 0 & \text{otherwise} \end{cases}$$

3.6 Output Parameters

MVA can provide only a limited set of output performance measures as compared to the GI/G/m analysis. As mentioned in the introduction to this chapter, the average values alone can be obtained by this algorithm. At the end of the iteration, we get the mean waiting time, queue size and arrival rate at each node. For Multiple class MVA, we get these parameters for each class of customers. In addition, the system throughput measured with respect to Node 1 is also obtained.

Chapter 4

Implementation Details

This chapter gives a brief outline of the implementation aspects of the *Network Analysis & Simulation Tool*. The main concern in the design of the package has been to develop a tool which can be handled with the least difficulty and has all provisions to satisfy the requirements of a typical performance analyst. At the same time, since the package is still in its development stage, care has to be taken to keep the program complexity at the minimum so that further modifications are easily possible. These two conflicting criteria have been taken into consideration in the development of the tool.

The software has been divided into two parts - the analysis program and the user interface program. In the following, we discuss these two sections.

4.1 The Analysis program

The analysis program consists of modules which execute the GI/G/m and the MVA algorithm. As is clear from the previous two chapters, the implementation of these algorithms on a high level language is not a feasible approach due to the large amount of computations involved. Hence, we have written the analysis program using a package called *Mathematica* which has been widely used for doing complex numerical calculations. Matrix computations of any order can be easily performed using *Mathematica* and therefore, this platform seemed an ideal choice for our purpose. In addition, *Mathematica* also allows symbolic manipulations which have been very helpful to us in solving the traffic rate equations in MVA. Another attractive feature is that *Mathematica* can be easily

interfaced to a C program, whereby we can execute *Mathematica* functions from within the C program. For this, *Mathematica* provides an “include” file called **mathlink.h** which contains the functions for interfacing. In fact, we can even call a C function from *Mathematica*, even though this provision has not been made use of in our software.

The core of *mathematica* is the **kernel**, which performs the computations. Interface to the kernel can be done through either the standard frontend provided by *Mathematica* or a user defined interface as in the case of the *Network Analysis & Simulation Tool*. The kernel to frontend connection is made through a high level communication standard called **Mathlink**.

An executable file is called as a **package** in *Mathematica* terminology and has an extension “.m”. The packages which have been developed for the G/G/m and MVA techniques are listed below.

ANALYSIS.M	the main package which calls the following packages
READGGMS.M	the package for reading the inputs for G/G/m Single class
READGGMM.M	the package for reading the inputs for G/G/m Multiple class
READMVAS.M	the package for reading the inputs for MVA Single class
GGM.M	the package for G/G/m algorithm (both Single and Multiple class
MVA.M	the package for Mean Value Analysis (Single class)

All the inputs are saved in a file called **in.txt** before *Mathematica* is called by the user interface program. The **ANALYSIS.M** package reads the set of inputs like number of nodes, type of solution algorithm etc. which is unique for any solution algorithm from this file. On the basis of the selections made by the user, the corresponding packages are called for reading the remaining inputs and then executing the algorithm. The outputs are stored in another file **out.txt** by the main package itself.

4.2 The User Interface program

The user interface program has been developed in C language using the Borland C++ Integrated Development Environment. It includes functions to create a network on the screen (i. e nodes and their interconnections), modify the network by addition/deletion

of nodes/lines, read the inputs and display the outputs and save/restore the network to/from a file. More details on how to use *Network Analysis & Simulation Tool* can be found in Appendix A. Note that only the PC version of the software has been developed.

Like any other Windows application, the *Network Analysis & Simulation Tool* interface makes use of the built-in graphical functions to do all the above mentioned operations. The user interface project file, **analysis.prj** includes the following files.

analysis.rc	the resource file for the project
analysis.def	the module definition file for the project
def.c	all the variable and function declarations
creatnet.c	functions for creation and modification of the network
display.c	functions for display of input and output parameters
getinput.c	functions for obtaining the input parameters from the user
phty.c	functions for obtaining the probability matrix entries
mathemat.c	functions for interface to <i>Mathematica</i>
reset.c	some general purpose functions like clear screen, reset nodes etc.

All input and output procedures, except those for the probability matrix, have been realised using the **Dialog Box** resource built in to Borland C++. This has simplified the code to a large extent and made further modifications easy. We have also taken into consideration memory problems, when there are a large number of nodes and classes. Due to a slight incompatibility in the manner in which numbers are handled by C and *Mathematica*, we had to store all the input and output parameters as strings, rather than real numbers. This may lead to lack of memory space, especially in cases when high precision is required. Hence, memory is allocated to nodes and classes whenever they are added to the existing network and is deallocated as soon as they are deleted. This helps to avoid any future problems when more elements may need to be added to the input and output sets.

4.3 C-*Mathematica* Interface

The interface between the C program and *Mathematica* is done as follows. As soon as the frontend of *Network Analysis & Simulation Tool* is started, *Mathematica* kernel is also launched through the program, and a link is established with it. When the user gives a command to begin analysis, all the inputs related to the network are stored in a file named “in.txt”. Next, we send a *Mathematica* command, “Get[analysis.m]” through the link which requests *Mathematica* to begin execution of the analysis program. In order to confirm that the analysis is completed, we send a small evaluation command (say “Plus[8,7]”) to *Mathematica* and then wait for the result to appear on the link. Since *Mathematica* processes instructions one by one, as soon as the result of this evaluation has been received, we can be sure that the analysis is completed and results have been stored in another file, “out.txt”. This extra check is required to avoid any complications which may arise if the user tries to view the output parameters at the same time when *Mathematica* is writing on to the output file.

Finally, the installation procedure for the *Network Analysis & Simulation Tool* is as follows. A program called “install.exe”, which is provided along with the software, is run. This program requests the user to specify the Install-from, Install-to and *Mathematica* directories. The install program creates the Install-to directory and then copies all the required files to it. A new executable file called “nast.exe” is created which is to be executed to start the front end program.

Chapter 5

Results and Conclusion

5.1 Results

In this chapter, we consider a few typical queueing networks and the results of their analysis using the *Network Analysis & Simulation Tool*. This has been done to demonstrate the range of applicability of this package and to check the accuracy of our results. Some of these examples have been taken from the references mentioned at the end of this report; others have been validated by direct evaluation.

Example 1

Here we analyse a single $M/E_3/1$ queue with an arrival rate $\lambda = 0.15$. The average service time is given as $\mu = 0.1$ and the coefficient of variation is 0.333. The queue has a multiplication factor of 1. The results of the analysis along with the values calculated using $M/E_r/1$ formulae are shown in Table 5.1.

Example 2

In this example, we have a 2-node Single class network analysed by Whitt[3], which is shown in Fig 5.1. Both are single server nodes, with multiplication factor 1. The external arrival process to Node 1 is hyperexponential with mean 1.00 and squared coefficient of variation 2.25. The mean service times at both the nodes are 1.00. The service time distribution at Node 1 is hyperexponential with squared coefficient of variation 2.25 and that at Node 2 is Erlangian-4 with squared coefficient of variation 0.25. The results are

Table 5.1: Output Parameters for Example 1

Parameter	NAST	$M/E_r/1$
Av.Number(system)	0.01515	0.0152
Sq.Coeff.Var.of Number(system)	67.2	66.3
Av.Wait.Time	0.00101	0.00099
Sq.Coeff.Var. of Wait.Time	298.1	296.7
Departure Rate,	0.15	0.15
Utilisation	0.015	0.015

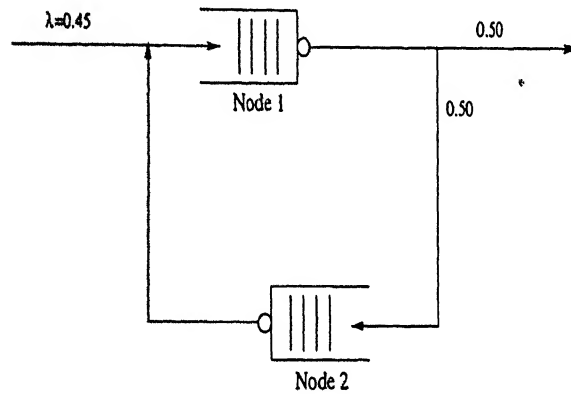


Figure 5.1: Example 2.

shown in Table 5.2.

The reference paper gives only the total sojourn time in the network, which is equal to 39.74. The corresponding value calculated by the *Network Analysis & Simulation Tool* is 40.1.

Example 3

Next we have a 4-node Single class queueing network as shown in Fig 5.2.

The input parameters are given in Table 5.3.

Table 5.2: Output Parameters for Example 2

Parameter	NAST	
	Node 1	Node 2
Av.Number(system)	17.1	0.79
Sq.Coeff.Var. of Number(system)	0.308	2.41
Av.Wait.Time	18.0	0.75
Sq.Coeff.Var. of Wait.Time	0.306	4.1
Departure Rate	0.45	0.0
Visit Ratio	2.00	1.00
Utilisation	0.90	0.45

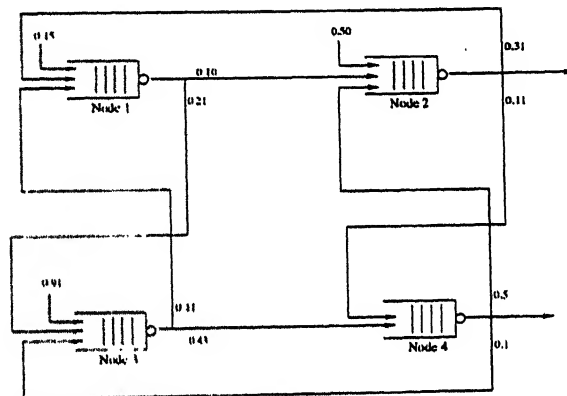


Figure 5.2: Example 3.

Table 5.3: Input Parameters for Example 3

Parameter	Node 1	Node 2	Node 3	Node 4
Ext.Arr.Rate	0.15	0.5	0.91	0.0
Sq.Coeff.Var.of Ext.Arr.	1.00	1.00	1.00	1.00
Mean Serv. Time	0.01	0.05	0.12	0.1
Sq.Coeff.Var.of Serv.Time	1.00	1.00	1.00	1.00
Mult.Factor	1	2	2	3
No. of servers	1	1	1	1

The results are shown in Tables 5.4 and 5.5. The internal arrival variance coefficients were found to be close to 1.00 and hence, we have also given the values which have been calculated by M/M/1 analysis on the network.

Table 5.4: Output Parameters for Example 3 (NAST)

<i>Parameter</i>	<i>NAST</i>			
	Node 1	Node 2	Node 3	Node 4
Av.Number(system)	0.076	1.11	0.96	1.36
Sq.Coeff.Var of Number(system)	14.0	1.45	2.01	1.58
Av.Wait.Time	0.00086	0.065	0.117	0.142
Sq.Coeff.Var of Wait.Time	21.2	1.59	2.97	2.07

Table 5.5: Output Parameters for Example 3 (M/M/1)

<i>Parameter</i>	<i>M/M/1</i>			
	Node 1	Node 2	Node 3	Node 4
Av.Number(system)	0.0753	0.927	0.953	1.279
Sq.Coeff.Var of Number(system)	12.8	1.38	1.93	1.77
Av.Wait.Time	0.00075	0.046	0.114	0.128
Sq.Coeff.Var of Wait.Time	19.8	1.23	3.45	2.15

Example 4

In this example, we consider a closed network of 4 queues with 10 customers of a single class circulating in it as shown in Fig 5.3.

Each node is a single server FCFS service centre and the mean service times are 0.02,0.03,0.042918 and 0.06132. The results of MVA analysis on this network are shown in Table 5.6.

For cross checking purpose, we have calculated the mean number and waiting time using another technique for Product Form closed networks, called Convolution Algorithm and the results were found to be matching.

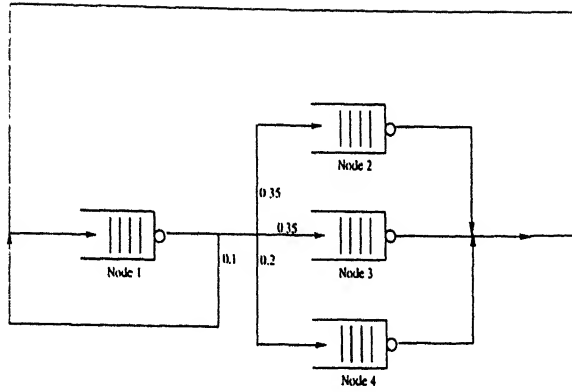


Figure 5.3: Example 4.

Table 5.6: Output Parameters for Example 4

<i>Parameter</i>	<i>NAST</i>			
	Node 1	Node 2	Node 3	Node 4
Av.Number(system)	5.5	0.97	2.17	1.33
Av.Waiting Time	0.095	0.0281	0.087	0.078
Visit Ratio	1.00	0.35	0.35	0.20

Example 5

We consider here another 4-node closed queueing network, with 20 customers as shown in Fig 5.4.

This is a modification of an example taken from [8] and models a central server system. Node 2 is an Infinite Server service centre and all the others process service requests in FCFS manner. The actual system consists of a CPU (Node 1), a set of terminals (Node 2), and two I/O devices (Node 3 and Node 4). The mean service time of these are 0.05, 20, 0.08 and 0.04 respectively. The results for this network, by MVA are shown in Table 5.7.

Example 6

We have a network of two nodes and three classes, which has been solved by the GI/G/m Multiple class algorithm. This example has been taken from [3] and only partial results are available. The number of servers in Node 1 and Node 2 are 40 and 10 respectively. The route inputs for the three classes is described below in Table 5.8.

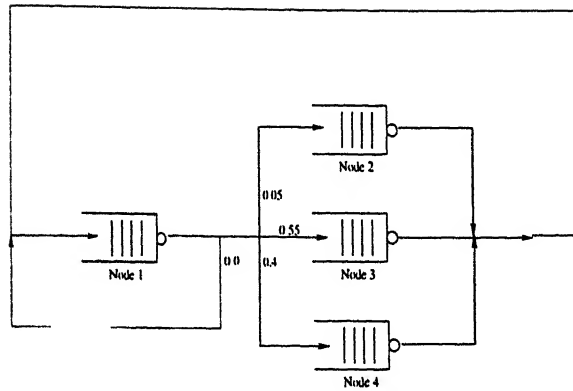


Figure 5.4: Example 5.

Table 5.7: Output Parameters for Example 5

Parameter	NAST			
	Node 1	Node 2	Node 3	Node 4
Av.Number(system)	2.47	15.4	1.77	0.32
Av.Waiting Time	0.110	0.0	0.129	0.0123
Visit Ratio	1.00	0.05	0.55	0.40

We give the results of analysis of this network using the *Network Analysis & Simulation Tool* in Tables 5.9 and 5.10.

In addition, the transition probability matrix is calculated from the route parameters and is shown in Table 5.11. The network is shown in Fig 5.5.

Example 7

Table 5.8: Route Parameters for Example 6

Route No.	Ext. Arr Rate	Sq. Coeff. Var. Ext. Arr.	Node Seq.	Av Serv Time	Sq. Coeff. Var. Serv. Time
1	2.0	1.0	1,1	1.0,3.0	1.0,3.0
2	3.0	2.0	1,2,1	2.0,1.0,2.0	0.0,1.0,1.0
3	2.0	4.0	2,1	1.0,2.0	1.0,2.0

Table 5.9: Output Parameters for Example 6 (Node Outputs)

<i>Parameter</i>	Node 1	Node 2
Av.Number(system)	24.0	5.0
Sq.Coeff.Var.Numb(system)	0.042	0.208
Av.Wait.Time	0.00033	0.0095
Sq.Coeff.Var.Wait.Time	530	31.3
Utilisation	0.60	0.50

Table 5.10: Output Parameters for Example 6 (Class Outputs)

<i>Parameter</i>	Class 1	Class 2	Class 3
Mean Wait.Time	0.00066	0.0102	0.0098
Mean Serv.Time	4.0	5.0	3.0

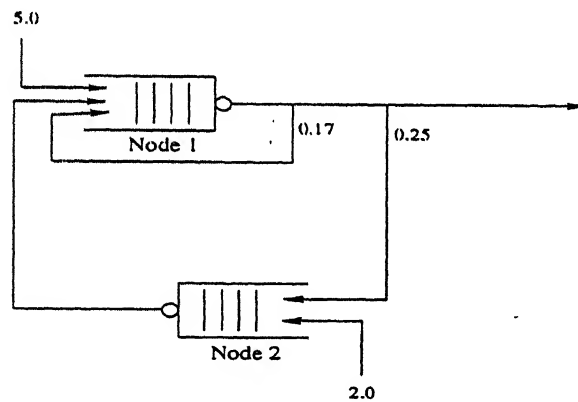


Figure 5.5: Example 6.

Table 5.11: Transition probability Matrix for Example 6

	1	2
1	0.167	0.25
2	1.0	0.0

As a last example, we consider a 12-node 20-class network, which has been analysed by Gelenbe and Mittrani[9]. All the nodes have constant service rates and hence the variation coefficient is zero. The external arrival process of each route has a squared coefficient of variation 4.0. Nodes 1,2,7,8,11 and 12 have mean service time equal to 0.2083 and all the other nodes have 0.02083. The route parameters of this network are shown in Table 5.12.

Table 5.12: Route parameters for Example 7

<i>Route No.</i>	<i>Ext. Arr. Rate</i>	<i>Node sequence</i>	<i>Route No.</i>	<i>Ext. Arr. Rate</i>	<i>Node sequence</i>
1	0.60	3	11	4.50	9
2	1.20	3,5	12	0.75	8
3	0.60	3,5,9	13	2.03	10,6,4
4	3.00	2	14	2.03	10,6
5	3.30	4	15	2.03	10
6	3.30	5	16	0.68	12
7	1.24	5,9	17	0.15	1
8	0.41	4,2	18	0.38	1,3
9	0.75	6,4	19	0.45	7
10	1.50	6	20	0.53	11

The results of GI/G/m Multiple class analysis on this network are shown in Tables 5.13 and 5.14. Note that all the output parameters have not been tabulated.

Table 5.13: Output Parameters for Example 7 (Node Outputs)

<i>Node No.</i>	<i>Utilisation</i>	<i>Av.Number</i>	<i>Av. Wait. Time</i>
1	0.110399	0.117	0.0129
2	0.835283	2.93	0.052
3	0.082903	0.060	0.00064
4	0.135187	0.146	0.00162
5	0.107066	0.142	0.00158
6	0.131437	0.141	0.00157
7	0.093735	0.098	0.0107
8	0.156225	0.170	0.0192
9	0.132062	0.142	0.00158
10	0.126855	0.136	0.00151
11	0.110391	0.117	0.0129
12	0.141641	0.153	0.0171

Table 5.14: Output Parameters for Example 7 (Class Outputs)

<i>Route No.</i>	<i>Av. Wait Time</i>	<i>Av.Serv. Time</i>	<i>Route No.</i>	<i>Av. Wait Time</i>	<i>Av.Serv Time</i>
1	0.001	0.021	11	0.003	0.021
2	0.002	0.042	12	0.077	0.208
3	0.006	0.063	13	0.006	0.063
4	1.918	0.208	14	0.004	0.042
5	0.002	0.021	15	0.002	0.021
6	0.002	0.021	16	0.069	0.208
7	0.005	0.042	17	0.026	0.208
8	1.920	0.229	18	0.026	0.229
9	0.004	0.042	19	0.043	0.208
10	0.002	0.021	20	0.052	0.208

5.2 Conclusion

In this thesis work, we have developed the framework of an integrated package for queueing network simulation and analysis. Two of the widely used solution techniques, viz. the GI/G/m and MVA algorithms have been incorporated in this package. The package has been tested on a wide variety of networks and has been performing well. Some of the results have been given in this chapter.

Other than the two techniques mentioned above, there is a large group of solution algorithms developed for different types of networks. Typically, networks with class priorities, capacity constraints[10] etc. , which are of much interest in performance modelling may be considered in the next step of modification of the package. In many cases, the probability distribution (exact or approximate) may be useful for the detailed analysis of a particular system. Techniques like the Diffusion Approximation[11] give the equilibrium distributions and can be integrated into this package.

The simulation part is proposed to be an independent software with the option of being used along with the *Network Analysis & Simulation Tool*. This seems to be a better approach as it would help to limit the complexity of the analysis section to a large extent.

Appendix A

How to use the *Network Analysis & Simulation Tool*

A.1 Create a simple network

This section gives a step by step procedure for creating a new network.

The shortcut keys for the selection of menu items are given in brackets.

Step 1: Selection of the Method of solution

The algorithm for solution can be decided based on the type of inputs you have and the variety of outputs which you require. Chapter 5 of this thesis report gives examples which may be useful in this context. Select the solution method using the *Options*(Alt-O) menu and then the *Method* submenu. If you are planning to simulate the network, select the *Simulation* submenu. Otherwise choose the *Analysis* menu and then the solution technique.

The default method is GGM.

Step 2: Selection of the type of input

Next you have to select the type of input which you have i. e Single or Multiple class. Select this using the *Options*(Alt-O) menu and then the *Input* submenu. Choose the type which you need from the popup window that is opened.

Step 3: Selection of the mode of creation

There are two methods by which you can create a network - *Auto* and *Man* modes. In the *Auto* mode the nodes and their interconnections are done automatically, where as in the *Man* mode you can manually do them. Select the *Tools*(Alt-T) menu and then the *Create* submenu.

It is suggested to use the Auto mode when the number of nodes is more than 15.

The *Auto* mode requires the number of nodes to be given prior to the creation of the network. A window pops up requesting you to enter the number of nodes.

Note: The GGM algorithm with Multiple classes of customers does not allow manual creation of the network. Hence, in case you have selected **GGM** in step 1 and **Multiple** in step 2, a window pops up requesting the no. of nodes and classes in your network. Then a prototype network is created on the screen with just the nodes shown without any interconnections.

Step 4: Creation of the network

This step varies depending on the selections made in steps 1 to 3.

1. GGM-Single-AUT:

The GGM method with a single class of customers requires the transition probability matrix to be entered before the network is created in the AUT mode. A new menu item *Pbty Mtx* is created as soon as you had done your selection in step 2. Select this menu (Alt-P) to enter the probability values. Once you have closed this window, the network is automatically created on the screen with a connection between two nodes made, if and only if the corresponding transition probability entry is non-zero. The colors of the lines drawn are decided by the software itself, in this mode of creation and you cannot modify them.

2. GGM-Single-MAN:

This mode of creation gives maximum flexibility to the user and is preferred in case the number of nodes is less. To create a node on the screen, move the cursor to the location where you want and then double click the left mouse button. Each node created has its default name displayed on it ("Node1", "Node 2" etc.) along with the identification number that is used for internal reference in the software. This

number is displayed on the probability matrix window as well. You can change the name of the node as you wish but this id number cannot be. Once you have created all the nodes you can terminate the node creation procedure by selecting the *Stop* (Alt-S) menu item. (This menu item is created only when you have selected MAN mode of creation in step 3 and disappears as soon as you have selected it.)

Next you may need to add the interconnections between nodes. For this, select the *Connect* (Alt-C) menu. You can also select the color with which you wish to draw the line, from a color palette which opens up if you select the *Colors* (Alt-R) menu. Now click the left mouse button ONCE on the node from which the connection starts and then on the one on which it terminates. A line would be drawn between these two and the corresponding probability matrix entry is made non-zero (to be entered later by the user). You can exit from the line creation process by selecting the *Stop* (Alt-S) menu as in the node creation. Once you have selected the *Stop* menu, the *Connect* menu item is disabled.

The probability matrix entries are dynamically modified as you create the network in the MAN mode with the entries for those nodes which are interconnected being kept blank and all other entries assigned to zero. You can select the *Pbty Mtx* menu and enter the values.

3. **GGM-Multiple:**

This mode of operation allows only auto creation of nodes and hence skips step 3. As soon as you have selected **Multiple** in step 2, a window pops up asking for the number of nodes and classes in the network. After you have entered these, the window closes and the nodes are created on the screen. No connections are made at this time, since the probability matrix is not to be given by the user and is later calculated from the set of inputs specified by him.

4. **MVA-Single:**

The MVA algorithm in Single class mode permits both MAN and AUT creation of nodes and is similar to those explained for GGM case. In addition, the algorithm also requires the total number of customers circulating in the closed network. You will be requested to enter this value, when you select *Single* option in step 3.

A.2 Modify the network

At some stage you may need to include one more node or line to the network that has been created as described in the last section. You may also need to delete a node or an interconnection. This can be done easily in this tool. The procedure varies slightly in the AUT and MAN mode and is independent of the type of solution and input selected.

Addition of a node:

AUT mode: To add a node in the AUT mode select the *Tools*(Alt-T) menu and then the *Modify-Add-Node* submenu. Now a window would pop up requesting the number of nodes to be added. Once you have entered this value, the *Pbty Mtx* menu appears highlighted indicating that the probability matrix has to be modified. Only after you have done this, the new nodes appear on the screen. During the addition of these extra nodes, the network is rearranged and the locations of some of the nodes may change. To make it easy for the user to identify where the new nodes have been positioned, they are highlighted, until the user selects them for input or output.

MAN mode: Addition of a node in MAN mode is done in the same way that nodes are created (i. e by left mouse button double click). For this you have to select the *Tools-Modify-Add-Node* menu. A *Stop* menu appears, selecting which you can leave the node-addition mode. The *Pbty Mtx* menu is highlighted indicating that it has to be modified.

Deletion of a node:

AUT mode: To delete a node in the AUT mode, select *Tools-Modify-Delete-Node* menu and then click ONCE on the node(s) to be deleted. The nodes and all its interconnections are removed from the screen and corresponding modifications are made in the probability matrix. You can terminate the deletion mode by double clicking the left mouse button. Again the *Pbty Mtx* menu is highlighted and has to be modified.

MAN mode: The procedure is the same as in AUT mode.

Warning: Do not forget to double click the left mouse button after you have deleted all the nodes which you need to.

Addition of a line:

AUT mode: The addition of a node in the AUT mode requires you to select the *Tools-Modify-Add-Line* menu. Next you have to click the left mouse button ONCE on the starting node of the line and then on the end node. The AUT mode addition of lines does not permit the user to select a line color of his choice; instead it uses a prefixed color on the basis of the starting node. Once you have completed with the addition of lines you can select the *Stop* menu to leave the line-addition mode. The *Pbty Mtx* menu is highlighted.

MAN mode: The procedure is the same as in the AUT mode. In this case, however, you can select the color of the line from the standard palette using the *Colors* menu.

Deletion of a line:

AUT mode: In the AUT mode to delete a line between two nodes, select the *Tools-Modify-Delete-Line* menu and then click ONCE each on the end points of the line to be deleted. To quit the line-deletion mode, double-click the left mouse button. The *Pbty Mtx* menu is again highlighted.

MAN mode: The procedure is the same as in the AUT mode.

Addition of a class:

AUT mode: To add a new class in the AUT mode, select the *Tools-Modify-Add-Class* menu item. The input window for the new class pops up. You may enter your inputs for that particular class in this window. Once the input window is closed, a selection window pops up which prompts you to select one of the following: **More**(Add one more class), **Close** (Exit from the window) and **Help**(Help). If **More** is selected, another input window for the new class is opened. This continues till you select **Close**.

MAN mode: This option does not exist in the GGM-Multiple case since no MAN mode is permitted.

Deletion of a class:

AUT mode: Select the *Tools-Modify-Delete-Class* menu item. An edit window is opened where you can enter the number of the class to be deleted. Once you have entered this,

the selected class is deleted and a message comes on the screen. The edit window remains on the screen until you close it.

MAN mode: This option does not exist in the GGM-Multiple case since no MAN mode is permitted.

A.3 Input/Output procedure

1. Node input/output

To give the input parameters for a node in the network, click ONCE on the node. A selection box is displayed from which you can select **Input**, **Output** or **Close**. Click on the **Input** button and a window is opened with the default values (or the present values in case you are modifying the input) of the input parameters for a node displayed. You can enter the new values in the edit box provided along with each parameter. The Tab key or the mouse can be used to move about in the window. You may enter the input values in any order. Note that only decimal numbers are allowed. After you have given all the required inputs you can close the window by clicking on the **Close** button.

To observe the output parameters of a particular node, click ONCE on the node and select the **Output** button this time. All the output parameters of the node for the particular solution algorithm are displayed on the screen.

2. Class input/output:

The class inputs can be given by selecting the *Class*(Alt-L) menu and then the *Input* submenu. A class selection window is opened where you can give the number of the class for which you wish to give inputs. Once you have entered this value, an input window for the class opens in which you can specify the input parameters for that class. When you press the **Close** button on this window, the initial class selection window again pops up and you can select a new class. This goes on till you close the selection window.

For class outputs select the *Class-Output* menu. An output window is opened giving all the output parameters for the particular class. On closing this, the selection window helps you to select a new class.

3. Probability Matrix:

Select the menu *Pbty Mtx*(Alt-P) to enter the probability values. A window pops up which gives the current probability values (default: 0.0). You can move about in this table using the keys : u(Up), d(Down), l(Left) and r(Right). You can quit this window by selecting the "X" button at the left hand top corner of the window or by pressing the q key.

4. Network output measures:

The output measures for the whole network can be observed by selecting the *Network* menu which is created as soon as the execution of the analysis/simulation program is completed.

A.4 Run the analysis / simulation program

Once all the inputs have been provided, you can start the execution of the *Mathematica* program for analysis/simulation by selecting the *Tools-Run* menu item. A window which displays "**Evaluating.....**" is opened and remains on the screen till the program ends. The user has to see to it that the inputs provided are valid and would not lead to any abnormal conditions.

Note: You may have to wait for a minute or even more (depending on the machine which you are using), since the loading of *Mathematica* takes up some time. It is advised not to do any further modifications during this period.

The results of the analysis, along with the inputs can be saved as a print file using the *Tools-Print Run Results* menu.

A.5 File operations

The *Network Analysis & Simulation Tool* has facilities for saving and restoring the all network details including input/output values and network configuration details. All these are included in the *File* menu.

To save the information, select the *File-Save* menu. In case you are saving it for the first time, a file selection box is displayed in which you can give the name of the file. The filename must have ".n" as its extension.

If you wish to save a network in a new name, select *File-Save As* menu and give the file name in the selection box displayed.

To clear the screen and begin with a new network select the *File-New* menu. The new network has a default file name "new.n"

To open a file for display , select the *File-Open* menu. The network information as saved is reproduced on the screen and you can continue modifying it.

Bibliography

- [1] F.Baskett et al, "Open,closed and mixed networks of queues with different classes of customers," *J. of ACM*, vol. 22, no. 3, pp. 248-260, April 1975.
- [2] Paul.J.Kuehn, "Approximate analysis of general queueing networks by decomposition," *IEEE Trans. Communications*, vol. 27, no. 1, pp. 113-126, January 1979.
- [3] Ward Whitt, "The queueing network analyser," *The Bell System Technical Journal*, vol. 62, no. 9, pp. 2779-2815, November 1983.
- [4] K.T.Marshall. "Some inequalities in queueing," *Operations Research*, vol. 16, no. 3, pp. 651-660, May-June 1968.
- [5] Ward Whitt, "Approximating a point process by a renewal process, two basic methods," *Operations Research*, vol. 30, no. 1, pp. 125-147, January-February 1982.
- [6] Susan L. Albin, "On poisson approximation for superposition arrival processes in queues," *Management Science*, vol. 28, no. 2, pp. 126-137, February 1982.
- [7] M.Reiser and S.S.Lavenberg, "Mean value analysis of closed multichain queueing networks," *J. of ACM*, vol. 27, no. 2, pp. 313-322, April 1980.
- [8] Kishor S.Trivedi, *Probability & Statistics with Reliability, Queueing and Computer Science Applications*, PHI, 1982.
- [9] E.Gelenbe and I.Mitrani, *Analysis and Synthesis of Computer Systems*, New York: Academy Press, 1980.
- [10] H.G Perros, "A bibliography of papers on queueing networks with finite capacity queues," *Performance Evaluation*, vol. 10, pp. 255-260, 1989.

- [11] Hishashi Kobayashi, "Application of the diffusion approximation to queueing networks in equilibrium queue distributions," *J. of ACM*, vol. 21, no. 2, pp. 316-328, April 1974.
- [12] Michael K. Molloy, *Fundamentals of Performance Modelling*, Macmillan Publishing Company, 1989.
- [13] Jim Conger, *Windows Programming Primer Plus*, Galgotia Publications Pvt. Ltd., 1995.
- [14] Stephen Wolfram, *Mathematica - A System for Doing Mathematics by Computer*, Addison Wesley, 1993.

123243

123243

This book is to be returned on the
date last stamped.

[illegible]

EE-1997-M-UME-WIN